Kernelized Linear Regression

We define a set of functions ϕ_i $i = 0, \dots, M$ (in total they are $M+1$) and consider such model:

$$y(x;w) = w^T \phi(x) \qquad \text{with } \phi(x) = \begin{bmatrix} \phi_0(x) \\ \vdots \\ \phi_M(x) \end{bmatrix} \quad \text{and } \phi_0(x) = 1$$

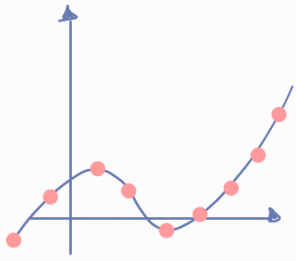
still M degrees of freedom

The resulting model is still linear in w but not in x (ϕ is usually non linear, otherwise this whole thing will be pointless).

e.g. we consider this particular transformation: $\phi = [x^0, x^1, \dots, x^M]^T$

$y(x;w) = w^T \phi(x) = (w_0, w_1, \dots, w_M)(x^0, x^1, \dots, x^M)^T = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M$

this is an M-degree polynomial



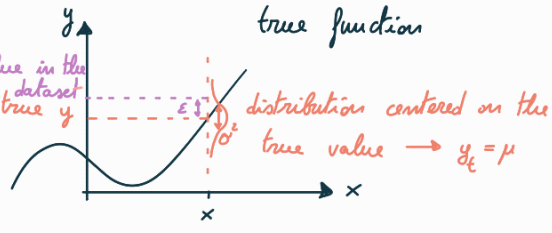
Solution of the problem is to find a good M such that we will best fit this data

← In this case we could chose $M = 3$

The choice of M is critical, it could lead to OVERFITTING: higher M could better fit the available data but we are interested in classifying new samples

Target values t in the dataset are given by the true function $y(x;w)$ affected by additive noise ε.

$$t = y(x;w) + \varepsilon$$



We assume that the noise follows a gaussian distribution.

We just described the same problem from a probabilistic point of view.

$$p(t \mid x, w, \beta) = \mathcal{N}(\varepsilon \mid y(x;w), \beta^{-1})$$

- likelihood
- the distribution is centered on the true value (zero-mean, we can translate it (?))
- variance is described in terms of precision β; precision is the inverse of the variance

Now we can try to maximize the likelihood (minimize the negative log likelihood)

$$\text{maximize} \quad p(\{t_1, \dots, t_N\} \mid x_1, \dots, x_N, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n \mid w^T \phi(x_n), \beta^{-1})$$

← under the hypothesis that the samples are independent and identically distributed we can use the theorem of total probability

or equivalently:

$$\text{minimize} \quad -\ln p(\{t_1, \dots, t_N\} \mid x_1, \dots, x_N, w, \beta) = \sum_{n=1}^{N} -\ln \mathcal{N}(t_n \mid w^T \phi(x_n), \beta^{-1})$$

$$= \underbrace{-\beta \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2}_{E_D(w)} - \frac{N}{2} \ln(2\pi\beta^{-1})$$

$E_D(w)$

do not depend on w

$E_D(w)$ is what we need to minimize

Maximum likelihood (minimum log likelihood) under zero-mean Gaussian assumption correspond to least square error minimization

$$\operatorname*{argmax}_w \; p(\{t_1, \dots, t_N\} \mid x_1, \dots, x_N, w, \beta) \longleftrightarrow \operatorname*{argmin}_w E_D(w) = \operatorname*{argmin}_w \frac{1}{2} \sum_{n=1}^{N} (t_n - w^T \phi(x_n))^2$$

sum of the squared errors: sum over all the points in the dataset of the squared difference between the true value t_n and the prediction $w^T \phi(x_n)$

Linear regression can be solved with this optimization function that derives at the same time from the probabilistic formulation of the problem (maximum likelihood) and from the least square minimization (squared difference between the true value t_n and the prediction $w^T \phi(x_n)$).

From the closed form solution:

① compute the gradient $E_D(w) = \frac{1}{2}(t - \phi w)^T (t - \phi w)$ ← square

with $t = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}$ $(N \times 1)$, $\phi = \begin{bmatrix} \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_M(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \dots & \phi_M(x_2) \\ \phi_0(x_N) & \phi_1(x_N) & \dots & \phi_M(x_N) \end{bmatrix}$ $(N \times M+1)$ $w = \begin{bmatrix} w_0 \\ \vdots \\ w_M \end{bmatrix}$ ← one weight for every element of the ϕ vector

② $\nabla E_D(w) = 0 \longleftrightarrow \phi^T \phi w = \phi^T t$

using M instead of $M+1$ for simplicity

③ isolate w from $\phi^T \phi w = \phi^T t$

$\phi^T \phi w = \phi^T t$

$(M \times N)(N \times M)(M \times \alpha) = (M \times N)(N \times 1)$

$(M \times M)(M \times \alpha) \qquad (M \times 1) \qquad \alpha = 1$

$$w = \boxed{(\phi^T \phi)^{-1} \phi^T} t$$

pseudo-inverse of $\phi \triangleq \phi^\dagger$

example

$f : \mathbb{R}^d \longrightarrow \mathbb{R}$

$D = \{ (x_n, t_n)_{n=1}^N \}$

define t, w and ϕ

we assume this particular transformation:

$$\phi = \begin{matrix} 1 \\ \\ \\ N \\ \uparrow \\ 0 \end{matrix} \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ & \vdots & & \\ 1 & x_N & x_N^2 & x_N^3 \end{bmatrix} \quad \uparrow M=3$$

the ϕ matrix will have N rows, one for each sample in the dataset and $M+1$ cols (number of features of the ϕ vector + ϕ_0

$$t = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix} \qquad w = \begin{bmatrix} w_0 \\ \vdots \\ w_M \end{bmatrix}$$

our model will be:

$y(x, w) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

$w_{ML} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} = \phi^\dagger t$ $\qquad$ $pinv(\phi) \cdot t$

w from maximum likelihood

We need an iterative algorithm. Why?

Because the ϕ matrix may be too large (N depends on the dataset size and M could also be very large)

Iterative algorithm: sequential learning

$\hat{w} \longleftarrow \hat{w} - \rho \nabla E_n$ ← learning rate

$\hat{w} \longleftarrow \hat{w} - \rho (t_n - w^T \phi(x_n)) \phi(x_n)$

this algorithm converges for suitable small values of ρ; its value need to be chosen carefully

Regression problem

$f : \mathbb{R}^d \longrightarrow \mathbb{R}$

$D = \{(x_n, t_n)_{n=1}^N\}$

for linear regression as well as for all other linear models, the functions we end up using for classification are linear in the weight but could (and should) be non-linear in the input

Linear regression

$y(x; w) = w_0 + w^T x \qquad x, w \in \mathbb{R}^d$

Linear regression with non-linear basis function

$y(x; w) = w^T \phi(x)$

$\phi = [\phi_0(x), \phi_1(x), \ldots, \phi_n(x)]^T \quad , \phi_0(x) = 1$

most commonly used basis function <
- polynomial
- radial
- sigmoid

Kernelized linear regression

linear regression is too simple for many applications
Start from a polinomial linear regressor :

$$y = w^T \phi(x)$$

$$\phi(x), w \in \mathbb{R}^n$$

$$\phi : \mathbb{R}^D \longrightarrow \mathbb{R}^n$$

polynomial regression ≠ non-linear regression

polynomial $\rightleftharpoons$ non-linear

$$y(x; w) = w^T \phi(x) = (w_0, w_1, \ldots, w_n)(x^0, x^1, \ldots x^n)^T$$

$\boxed{\phi_n(x) = x^n}$

How do we find the best w ?
Define a cost function :

$$J(w) = \sum_{n=1}^{N} (y_n - w^T x_n)^2$$ least squares cost function

Minimize the cost function with respect to the weights

$$w^* = \arg\min_w J(w)$$

After minimizing and simplifying we get :

$$w = (X^T X)^{-1} X^T Y$$

For the non-linear basis function :

$$w = (\phi^T \phi)^{-1} \phi^T Y$$

With regularization :

$$w = (\phi^T \phi + \lambda I)^{-1} \phi^T Y$$

with regularization we talk about Ridge Regression

Estimating the value of w in terms of ϕ is very computationally expensive ($\phi^T\phi$) but there is another way (Kernelization)
Lets start again with the Ridge Regression cost function

$$J(w) = \sum_{n=1}^{N} (y_n - w^T\phi(x_n))^2 + \frac{1}{2}\lambda \sum_{n=1}^{N} \|w\|^2$$

Take the derivative with respect to the weight vector and equate it to zero

$$w^* = \frac{1}{\lambda} \sum_{n=1}^{N} (y_n - w^T\phi(x_n))\,\phi(x_n)$$

$$w^* = \sum_{n=1}^{N} \underbrace{\frac{1}{\lambda}(y_n - w^T\phi(x_n))}_{\alpha_n}\,\phi(x_n)$$

$$w^* = \sum_{n=1}^{N} \alpha_n \phi(x_n)$$

$$= \phi^T \alpha \quad \text{dot product notation}$$

Go back to the original cost function and vectorize w

$$J(w) = \sum_{n=1}^{N} (y_n - w^T\phi(x_n))^2 + \frac{1}{2}\lambda \sum_{n=1}^{N} \|w\|^2$$

$$J(w) = (y - \phi w)^T(y - \phi w) + \frac{\lambda}{2} w^T w$$

Simplify

$$J(w) = y^Ty - y^T\phi w - w^T\phi^T y + w^T\phi^T\phi w + \frac{\lambda}{2} w^T w$$

$$J(w) = y^Ty - y^T\phi(\phi^T\alpha) - (\phi^T\alpha)^T\phi^T y + (\phi^T\alpha)^T\phi^T\phi(\phi^T\alpha) + \frac{\lambda}{2}(\phi^T\alpha)^T(\phi^T\alpha) \qquad w = \phi^T\alpha$$

$$J(w) = y^Ty - y^T\phi\phi^T\alpha - \alpha^T\phi\phi^T y + \alpha^T\phi\phi^T\phi\phi^T\alpha + \frac{\lambda}{2}\alpha^T\phi\phi^T\alpha$$

$\phi\phi^T$ is called Gram matrix / Kernel matrix (K)

$$\phi\phi^T = K = \begin{bmatrix} \phi(x_1)^T\phi(x_1) & \phi(x_1)^T\phi(x_2) & \cdots & \phi(x_1)^T\phi(x_n) \\ \phi(x_2)^T\phi(x_1) & \phi(x_2)^T\phi(x_2) & \cdots & \phi(x_2)^T\phi(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(x_n)^T\phi(x_1) & \phi(x_n)^T\phi(x_2) & \cdots & \phi(x_n)^T\phi(x_n) \end{bmatrix}$$

This matrix is significant because of its two properties:
① Symmetry: $K = K^T$
② Positive semi-definite: $\alpha^T K \alpha \geq 0$ (product with any other vector and its transpose is ≥ 0)

Let's use these properties in our simplification of the cost function

$$J(w) = y^Ty - y^T\phi\phi^T\alpha - \alpha^T\phi\phi^T y + \alpha^T\phi\phi^T\phi\phi^T\alpha + \frac{\lambda}{2}\alpha^T\phi\phi^T\alpha$$

$$J(w) = y^Ty - y^TK\alpha - \alpha^TKy + \alpha^TK^2\alpha + \frac{\lambda}{2}\alpha^TK\alpha \qquad \because K = K^T$$

$$J(w) = y^Ty - y^TK\alpha - y^TK\alpha + \alpha^TK^2\alpha + \frac{\lambda}{2}\alpha^TK\alpha \qquad \text{transpose the scalar terms}$$

$$J(w) = y^Ty - 2y^TK\alpha + \alpha^TK^2\alpha + \frac{\lambda}{2}\alpha^TK\alpha$$

$$\frac{\partial J(\alpha)}{\partial \alpha} = 0 - 2y^T K + \alpha^T K^2 + \frac{\lambda}{2}\alpha^T K$$

$$-2y^T K + \alpha^T K^2 + \frac{\lambda}{2}\alpha^T K = 0$$

$$K\left(-2y^T + \alpha^T K + \frac{\lambda}{2}\alpha^T\right) = 0$$

$$-2y^T + \alpha^T K + \frac{\lambda}{2}\alpha^T = 0$$

$$\alpha^T K + \frac{\lambda}{2}\alpha^T = 2y^T$$

$$\alpha^T\left(K + \frac{\lambda}{2}I\right) = 2y^T$$

$$\alpha^T = 2y^T\left(K + \frac{\lambda}{2}I\right)^{-1}$$

$$\alpha = \left(K + \frac{\lambda}{2}I\right)^{-1} 2y \qquad \text{we can drop the scaling factor 2}$$

$$\alpha^* = \left(K + \lambda' I\right)^{-1} y$$

We previously had:

$$w = \phi^T \alpha \longrightarrow w = \phi^T\left(K + \lambda' I\right)^{-1} y$$

Not kernelized:	$w = \left(\phi^T\phi + \lambda I\right)^{-1}\phi^T y$
Kernelized:	$w = \phi^T\underbrace{\left(\phi\phi^T}_{K} + \lambda' I\right)^{-1} y$

$\phi\phi^T$ is easier to compute than $\phi^T\phi$

According to Mercer's theorem, a symmetric function $k(x,y)$ can be expressed as an inner product $k(x,y) = \langle \phi(x), \phi(y)\rangle$ for some ϕ if and only if $k(x,y)$ is positive semidefinite, i.e.

$$\int k(x,y)\, g(x)\, g(y)\, dx\, dy \geq 0 \quad \forall g$$

or equivalently:

$$\begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots \\ k(x_2,x_1) & \ddots & \\ \vdots & & \end{bmatrix}$$ is positive semidefinite for any collection $\{x_1, \ldots, x_n\}$

Thanks to Mercer's theorem, we can conclude that we can rewrite every term in the Gram matrix K as a function $k(x_i, x_j)$

① Symmetry: $K = K^T$
② Positive semi-definite: $\alpha^T K \alpha \geq 0$

$\longleftrightarrow$ $k(x,y) = \langle \phi(x), \phi(y)\rangle$

we had the inner product, we switch to the function instead

$$\phi\phi^T = K = \begin{bmatrix} \phi(x_1)^T\phi(x_1) & \phi(x_1)^T\phi(x_2) & \cdots & \phi(x_1)^T\phi(x_n) \\ \phi(x_2)^T\phi(x_1) & \phi(x_2)^T\phi(x_2) & \cdots & \phi(x_2)^T\phi(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(x_n)^T\phi(x_1) & \phi(x_n)^T\phi(x_2) & \cdots & \phi(x_n)^T\phi(x_n) \end{bmatrix} = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \cdots & k(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{bmatrix}$$

Consider this polynomial non-linear basis function

$$\phi : x \longrightarrow \phi(x) = \begin{bmatrix} x_1^2 \\ \sqrt{2}\, x_1 x_2 \\ x_2^2 \end{bmatrix}$$

We have a kernel matrix where every term is an inner product of two samples

$$\phi(x_m)^T \phi(x_n) = \begin{bmatrix} x_{m1}^2 \\ \sqrt{2}\, x_{m1} x_{m2} \\ x_{m2}^2 \end{bmatrix}^T \begin{bmatrix} x_{n1}^2 \\ \sqrt{2}\, x_{n1} x_{n2} \\ x_{n2}^2 \end{bmatrix}$$

$$= x_{m1}^2 x_{n1}^2 + 2 x_{m1} x_{m2} x_{n1} x_{n2} + x_{m2}^2 x_{n2}^2$$

$$= (x_{m1} x_{n1} + x_{m2} x_{n2})^2$$

$$= (x_m^T x_n)^2$$

$$= k(x_m, x_n)$$

Expanding the product we can see that it can be represented as a function of the basis vector inputs x_m and x_n

This in particular is an example of a polynomial kernel function
We can compute the kernel matrix k without knowing the true nature of ϕ, only knowing the kernel function k

How can we make predictions ?

$$y(x) = w^T \phi(x) \qquad w = \phi^T (k + \lambda' I)^{-1} y$$

$$= y (\phi \phi^T + \lambda' I)^{-1} \phi^T \phi(x)$$

y in the dataset

we can express $\phi^T \phi(x)$ as a column vector of the kernel matrix

$$\phi^T \phi(x) = \begin{bmatrix} \phi(x_1)^T \phi(x) \\ \phi(x_2)^T \phi(x) \\ \vdots \\ \phi(x_n)^T \phi(x) \end{bmatrix} = k_x$$

$$= y (k + \lambda' I)^{-1} k_x \qquad \text{independent of } \phi$$

Even with complex basis functions we can make predictions with only the base features